Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 44 (2015) 578 – 587

Procedia Computer Science

2015 Conference on Systems Engineering Research

PARADIGMshift: A Method for Feasibility Studies of New Systems

Holger Schumann*, Axel Berres, Sönke Escher, Tilman Stehr

German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany

Abstract

A conceptual design method to perform feasibility studies for new technical systems is presented. The method enforces a formal yet pragmatic definition of the problem domain, the solution candidates, and the way of candidate selection for deeper analyses. General cross-cutting development approaches like simulation and function-based design are evaluated regarding their applicability for feasibility studies. Suitable concepts are identified. Based on application examples from different domains like space and aerospace, it is shown how the PARADIGMshift method realizes the identified beneficial concepts for feasibility studies.

Peer-review under responsibility of the Stevens Institute of Technology.

Keywords: Conceptual Design; Conception; Feasibility; Trade-Off;

1. Introduction

International competition forces all manufacturers of new systems to increase the innovation rate and to shorten the time-to-market. Besides a high product quality, efficient development processes are required to achieve these goals. The conception stage[1] represents the first development stage. It consists of one or more short-time studies to estimate e.g. the technical, commercial, legal, and financial feasibility of a new system and its variants. Based on the results of such feasibility studies, trend-setting and cost-determining decisions are made. About 70% of the total life-cycle cost is determined in the conception stage even though no detailed information and analyses are available[2].

* Corresponding author. Tel.: +49-531-295-2776; fax: +49-531-295-2931.
E-mail address: holger.schumann@dlr.de

Peer-review under responsibility of the Stevens Institute of Technology.
doi:10.1016/j.procs.2015.03.023

This fact highlights the importance of the conception stage and provides motivation for a closer look into the methods used at that stage.

What makes conceptual design as well as feasibility studies special is that they are performed early in the development process within a short period of time in comparison to the overall design effort. A system-of-interest considered in a conceptual design study introduces new challenges. Either the system has never been built this way before or new technologies or engineering disciplines are needed. To manage these new challenges, the composition of the design team is usually changed. This newly arranged team does not have the time to learn a comprehensive design method or to become familiarized with new collaboration or design tools.

In consequence, the team members will neglect to follow a design method. Instead, they will use ad-hoc tools well-known to them, such as Excel, Powerpoint, or mind mapping tools. This tool-driven ad-hoc approach is sometimes jokingly called "Powerpoint Engineering", as those unspecialized tools are originally not intended to be used for requirements engineering or systems design. For example, they usually do not support hierarchical decompositions or traceability. Their lack of formality complicates the maintainability and reusability of their data outputs and the interoperability with discipline-specific engineering tools. Additionally, the seamless transition to the next development stage is limited.

To overcome this problem, a pragmatic and efficient method specialized for feasibility studies is needed which is supported by an intuitive design tool. According to the development paradigm "process drives tools"[3], the method has to be defined prior to the development of the supporting tool.

In this paper, existing general design methods and concepts are presented and, based on that, a specialized method for feasibility studies called PARADIGMshift is introduced. Its implementation, application, and evaluation are presented.

2. Systems Design Methods

A detailed analysis of all the design methods currently applied worldwide is hardly possible. However, descriptions of top-level methods and concepts are available which can be used to outline the state of the art. The INCOSE handbook lists common development methods that can be used across the whole system's life-cycle. In the following, the Quality Function Deployment (QFD) method and the "Cross-Cutting Technical Methods" described in the handbook are used as a guideline to summarize the methods available for feasibility studies.

2.1. Quality Function Deployment (QFD)

QFD[4] is a quality assurance technique and provides a fast and systematic way to translate the customer's demands, also called voice of the customer (VOC), into design parts and characteristics for e.g. manufacturing and production. By means of a set of matrices, any correlations between characteristics and quality aspects of processes, requirements, design parts and more can be represented. One important matrix juxtaposes the customer's demands with the features of the product to be built. If this matrix is extended on top by a triangular matrix correlating feature requirements to themselves and at the bottom by a table of benchmark criteria, it is called "The House of Quality". This "house" helps to reveal unmet requirements as well as unnecessary features and to highlight the important ones.

2.2. Modeling, Simulation, and Prototyping

Modeling as well as simulation represent a specific view on the systems-of-interest by giving insight into the electrical characteristics or the mechanical behavior of the system, for example. Those methods are used to align the developers' understanding and expectations with physical laws and to ease the communication among the team. Besides, they do not only help to identify risks early and to avoid faults and their costly impact on later life-cycle stages, but also help to identify key technologies, key parameter sensitivities, and dependencies between the system's elements. Finally, they support decision-making by predicting the system's behavior. In conjunction with this method, the following terms should also be considered:

- Multidisciplinary Design and Simulation: Expresses explicitly that there are multiple engineering disciplines involved, which implies that multiple human experts with different backgrounds participate in all of the design and simulation tasks. If all related disciplines are participating, this approach enables a holistic view on the system, providing a fundamental base for the quality of feasibility studies.
- Concurrent Engineering or Concurrent Design[5,6]: Represents a multidisciplinary approach and expresses explicitly that the experts for engineering and operation, including the customer, are negotiating the optimal system configuration simultaneously. To overcome the high communication effort, they are typically located at one place, e.g. a conference center, and organize their design tasks in multiple sessions, often within a few weeks only. To support efficient information exchange and tool interoperability, the data exchange is mainly based on parameter sets. Typically, there are new team members participating, and due to lack of time, the infrastructure and tools to be used must first and foremost be clear and intuitive. The Concurrent Engineering approach is essential for high efficiency of feasibility studies.

2.3. Functions-Based Systems Engineering Method

A system function represents what the system will do and not how it will do it. The top-level functions express the customer's as well as the users' needs. The purpose of the method is to identify all necessary system functions and to provide foundation for the allocation and traceability of designed system elements to the system functions, comparable to the QFD approach. An aileron of an aircraft, for example, can be allocated to the roll function of flight control.

To achieve that, the top-level functions are hierarchically decomposed until basic sub-functions and their inputs, outputs, dependencies and constraints are described. This also includes the functions' usage modes, their internal as well as external interfaces, and their input, output, and performance requirements. This method typically iterates with the process of "Stakeholder Requirements Definition"[1] until the previously defined stop criteria are reached. The resulting functional architecture is typically expressed by IDEF0[7] or SysML[8] diagrams. These diagrams visualize the architecture or alternative decompositions and foster decision-making among the team. The functional requirements and constraints can be seen as design criteria for any solution candidates.

Besides Function-Based, this method is also known as Function-Driven as well as Function-Oriented Design. The corresponding FAS method[18] creates functional architectures for systems with SysML and is further developed by the related working group of the German INCOSE chapter.

Because functions have a longer period of validity than technological implementations, the Function-Based approach improves the chances for reuse of feasibility study results. Furthermore, describing functions in advance of formulating requirements allows the pragmatic step of describing what the system will do. The step of formulating detailed requirements can be shortened in favor of feasibility study efficiency without omitting fundamental functionality.

2.4. Object-Oriented Systems Engineering Method

The method is called OOSEM in short. It is a top-down approach, which means that engineers start from designing high-level system functions or elements and end with detailing basic sub-functions or elements. This decomposition is practiced until the level of detail is reached which was previously specified or which allows the full implementation of the system elements. The top-down approach enables a clear system overview from the beginning and enhances the overall understanding of the system, which eases planning and integration of elements. OOSEM considers specification, analysis, design, and verification of systems, which supports a holistic view on the system. In this way, it accommodates evolving technologies and changing requirements and provides the preconditions for a seamless transition through the conception and development[1] stages. It can be seen as an elaboration of the technical processes defined in ISO/IEC 15288[1]. There is a corresponding INCOSE working group providing more detailed information about this method. The following concepts are also considered by this method:

- Model-Based Systems Engineering (MBSE): The core of this concept is one system model storing all information relevant for one or more engineering disciplines. The model, also called Integrated Data Model[9],

accompanies the system's life-cycle. Because the system model represents the dependencies between the engineering disciplines, it builds the base for an efficient interdisciplinary communication and a better common understanding. In this way, the concept improves the quality of the design, reduces its risks, and helps to master the complexity of the system-of-interest. It replaces the common documents-centric development approach by a model-centric one[10].

- Systems Modeling Language: A modeling language describes a model using a defined notation. The most widely spread systems modeling language is SysML[8]. The language was specified by OMG (Object Management Group) and provides a graphical notation in terms of several diagrams. The set of diagrams is comprised of several structure and behavior diagrams, a requirement diagram, and a parametric diagram. Diagrams greatly foster the interdisciplinary communication among design teams.

2.5. Desirable Properties for a Design Method Supporting Feasibility Studies

A design method specialized for feasibility studies should incorporate the concepts described above to share in their benefits. But the specialty of feasibility studies in terms of having new teams and limited resources for learning prevents the full and unconditional integration of the concepts. Nevertheless, a new approach should be:

- Top-Down and Multidisciplinary: To provide a clear and holistic overview of the system and to disclose interdisciplinary dependencies. This, in turn, helps to improve the team's understanding, to identify risks, and to master the system's complexity. The chances for a successful integration of elements are increased and decision-making is supported. Can be efficiently applied by specifying a maximum decomposition depth. If there is a systems engineer guiding the team in top-down thinking, no mentionable learning effort for the team is needed here.
- QFD- and Function-Based: To provide a pragmatic way of defining user needs instead of formal writing of requirements. Function descriptions help to identify key technologies and increase the chances for the reuse of study results due to their independence from technology. Compared to specifying correct requirements, the effort for defining system functions and for learning how to do this is lower due to a higher abstraction level. For example, and according to Airbus, the flight control functions of a passenger aircraft can be defined by a few dozens of functions. On the contrary, the related formal requirements set for the flight control system contains significantly more entries.
- Model-Based: A system model or Integrated Data Model improves the formality and quality of the design. It enables a seamless transition to next stages. The model should be defined prior to the feasibility study and should be assembled by a few common data types only to reduce the learning effort. It is difficult to implement a detailed simulation model for predicting the system's dynamic behavior within the short-time period that is usually available for the study. Instead, the system model could depict the system's behavior in a discrete way in terms of hierarchical state machines[11] which may be executable. This kind of simulation may enhance decision-making in a quite pragmatic way.
- Concurrent and Parameter-Based: To improve the efficiency of feasibility studies. To achieve this efficiency, all information that need to be exchanged between the different disciplines may be expressed by parameters, which eases tool interoperability. Additionally, clear and intuitive tools requiring a minor learning effort are needed to support this collaborative approach.
- Graphically Representing: To visualize the functional as well as system architecture and the element's dependencies. This improves the interdisciplinary communication and decision-making. If SysML is not known to all team members, the learning effort for using SysML for modeling is quite high. Nevertheless, diagrams are needed but they may be generated from the system data model instead of being drawn manually.

At DLR, a method called PARADIGMshift has been developed to support feasibility studies by incorporating the design concepts as described above.

3. Design Method PARADIGMshift

The method has been developed by DLR to provide a specialized design method for feasibility studies. It is inspired by the approaches described above. But in contrast to them, it focuses on the special needs of the early conceptual design phase. To satisfy those needs, it integrates the different methods into one coherent whole. To separate thinking about the user needs, called the problem domain, from thinking about solutions and their trade-off, the method consists of three parts as described in the next sections.

3.1. Problem Domain

First of all, the user needs derived from a business case are stated in form of system functions like “The aircraft provides transport capacity”. The functions may be further decomposed to sub-functions like “It provides passenger transport” and “It provides cargo transport”. The sub-functions may be further specified by formal requirements, e.g. to determine the kind of cargo. In any case, the minimum number of passengers is formally defined by a parameter constraint like “minPax=150” attached to the function. This constraint is used to validate the related parameter value in the solution models afterwards. Additionally, key parameters such as the number of passengers are identified in this way.

In some cases, the constraint may depend on specific system states or aircraft states respectively. The required roll performance, for example, depends on the states “Landing” or “Take-Off”. To consider this, all relevant system states regarding the problem domain are separately defined by a simple list or more detailed by a hierarchical state machine[11]. States may also include environmental states like “Crosswind” and “Wind Gust”. Fig. 1 shows an exemplary Integrated Data Model representing the aircraft functions and the environment (atmosphere) definition. For the aircraft, some states (flight phases), functions, and sub-functions with parameter constraints are defined.

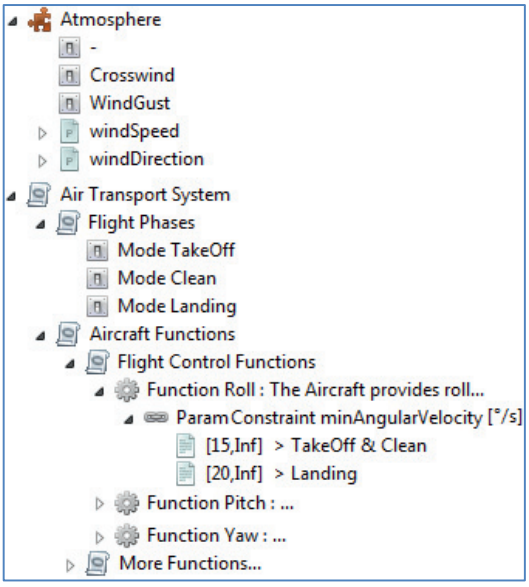

Figure 1. Integrated Data Model representing the problem domain of a system in terms of environment and aircraft functions

Dependencies between functions can be modeled and visualized by an exported diagram. Evaluation criteria are defined based on the identified key parameters. For every key parameter, evaluation weights or optimal values may be defined to support the trade-off between different solution candidates. In summary, the steps in the problem domain are:

- Definition of the functional architecture including decomposed use cases, system functions, and requirements. Additionally, definition of the environment.
- Definition of states (top-level operating modes and environmental states).
- Identification of key parameters and specification of their value constraints.
- Definition of evaluation criteria.

The Integrated Data Model, as exemplarily shown, stores all user requirements mainly in terms of functions and parameter constraints. It provides this information at any time to all team members in a formal, consistent, and traceable way. The functions and their constraints are now prepared for being assigned to components of the design solutions.

3.2. Solutions Domain

In this domain, several solution candidates are defined and the components and parameters of every solution are assigned to the functions and constraints of the problem domain. This procedure reveals every unconsidered function and ensures full bidirectional traceability between all functions and solution components.

To define a solution, logical components are identified and defined first. Such top-level components can be seen as containers for a group of components like a propulsion system containing engine, gear drive, and fuel tank components among others. While decomposing and describing the logical and physical components, their dependencies and physical connections to other components are specified. The system architecture is the result of this first design step.

In the next step, the architecture is enhanced by an approximate description of the system's behavior in terms of system states or operation modes. The states defined in the problem domain can be reused or new more detailed ones can be created. This may include failure states, for example. The transitions between the states are defined and transition constraints are specified. An aircraft, for example, can have state transitions from "on ground" to "airborne" and vice versa. For the landing gear, a transition constraint may be defined because it can only switch to "retracted" if the aircraft is in the air. In this way, a hierarchical state machine is defined to reflect the system's behavior in a discrete way.

The components are further detailed and formally described by the specification of parameters and their values. In addition to the system's performance, the parameters may also represent rating values for expected development as well as operating costs, producibility, maintainability, usability and so on. The cruise speed, for example, may be a performance parameter of an aircraft system. The parameter values may be estimated or taken from fact sheets and inserted manually. They may also be calculated on the basis of other parameters such as engine performance, total weight, and aerodynamic drag. All parameter values and calculation functions are transparently stored in an Integrated Data Model. Existing components that can be reused and integrated may be described the same way. To enable a kind of interoperability with those components, their descriptions may be stored in a catalogue which may be accessed by the Integrated Data Model.

Finally, key parameters are assigned to a related parameter constraint from the problem domain to allow their automated validation. As long as all parameter constraints are satisfied, changing parameter values now enables sensitivity or what-if analyses.

In summary, the following steps are performed for every solution candidate:

- Definition of the system architecture comprising logical and physical components and its dependencies.
- Definition of hierarchical state machines including transitions and constraints.
- Definition of performance parameters and their values or calculation functions.
- Assignment of parameters to parameter constraints, validation of parameters and parameter optimization.

Fig. 2 shows an exemplary Integrated Data Model representing a solution candidate. In the depicted model, the decomposition of some logical components and a parameter "rollAngularVelocity" are shown. This parameter is assigned to the parameter constraint from the problem domain and satisfies it (assignment not depicted).

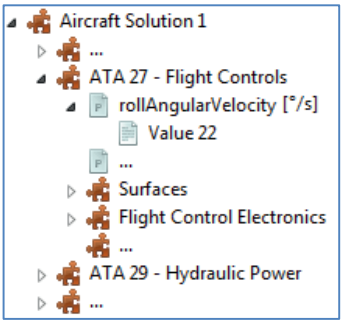

Figure 2. Integrated Data Model representing a solution candidate for an air transport system

Typically, several solution candidates are defined to reflect all of the engineers' solution ideas. The candidates may differ e.g. in their system architectures or only in their initial parameter values. The 2 or 3 most promising candidates have to be selected for deeper analysis and design to reduce the overall development effort.

3.3. Trade-Off Analysis

This analysis represents a pragmatic kind of decision-making providing reproducible decisions predominantly based on data instead of subjective impressions. The evaluation criteria are already defined during the problem definition step. The criteria include the key parameters, the parameters' relevance in terms of weights, and their optimal values. Based on this data, an evaluation value is calculated to enable a ranking of the solution candidates. A simple but useful calculation function may be:

$$E = \sum_{i=1}^{n} \left(\left(1 - \frac{abs(P_i - P_{i,opt})}{P_{i,\max}} \right) * P_{i,weight} \right)$$

with E: evaluation value; n: count of key parameters; P_i: value of parameter *i*; $P_{i,opt}$: its optimal value; $P_{i,max}$: maximum nonzero distance between P_i and $P_{i,opt}$ over all solutions; $P_{i,weight}$: evaluation weight of P_i

The trade-off analysis may also lead to the conclusion that some of the defined system functions or designed components have to be improved or specified in more detail to allow a meaningful assessment. In this way, one or more design iterations may be necessary.

4. Application and Evaluation

The method is strongly related to the underlying Integrated Data Model and the modeling tool. Its application is hardly possible if the general structure of the system model, the so-called metamodel, is not defined. Similarly, modeling the system requires a proper modeling tool especially when efficient working is intended. For this reason, DLR has developed an Integrated Data Model called PREMISE[14] and related modeling tools. These technologies can be seen as reference implementations which are not in the focus of this paper. Even though the papers referenced in this section mainly address the technological issues, they include the application examples for the PARADIGMshift method. Although not explicitly mentioned, the method was implicitly or partly applied.

4.1. Space

DLR has recently launched a space mission for an in-situ examination of an asteroid. During the mission's planning, different versions of the small landing device called MASCOT (Mobile Asteroid Surface Scout)[12] were designed in several design and concurrent engineering studies. Before the introduction of Concurrent Engineering, designing space systems took at least a few months due to a mainly sequential design process and engineers working at distributed locations. Now, design alternatives for the MASCOT lander, for example, are created within 2 weeks at the DLR Concurrent Engineering Facility. The systems engineer leads a multidisciplinary team which develops top-down architectures, operating modes, and related system parameters such as launch mass and power consumption. The learning effort for team members to become familiar with the collaboration method, the data model and tool has been reduced to 2 hours. In the early studies, the data model and tool were mainly Excel-based. It was argued[13] that the design approach could be further improved by the use of a more specialized concurrent design tool called "Virtual Satellite" developed by DLR. It was shown[14] that the MASCOT study could be successfully reproduced using the new tool.

After all, the high efficiency of the design method has been achieved because of the method's top-down, multidisciplinary, model-based, and concurrent character which was required in section 2.5 and which is now part of PARADIGMshift.

4.2. Aerospace

Until 2012, DLR operated the last version of the VFW 614 passenger aircraft (Fig. 3a) which served as an advanced technology demonstrator (ATD). To allow its use as a flying simulator, the original flight control configuration was extended by a modern electronic flight control system (EFCS). This EFCS has been used as an example to demonstrate the practicability of the PARADIGMshift method and the PREMISE data model.

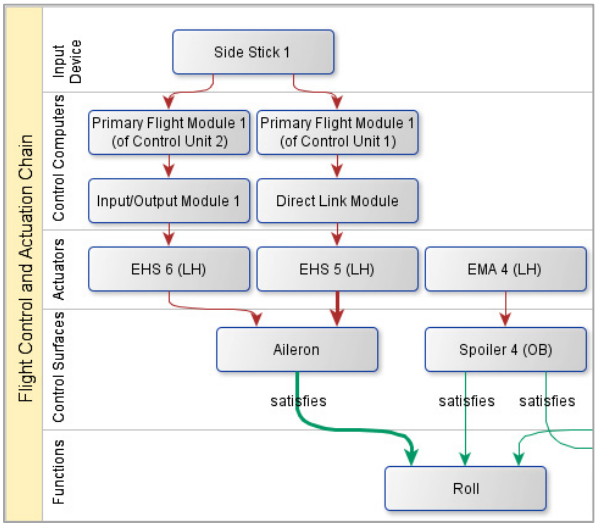

Figure 3. (a) VFW 614 ATD operated by DLR; (b) Excerpt of the generated architecture diagram

The resulting EFCS model was divided into a problem part and a solution part. The problem part contained the top-level flight control functions, and the solution part was comprised of the logical and physical components which were assigned to the related function. The engineers agreed that this top-down functional approach was advantageous to identify all of the needed functions without getting caught in discussions about technical solutions. Fig. 3b represents an excerpt of the architecture diagram[17]. The bottom part exemplarily shows the correlation between the roll function and physical control surfaces like aileron and outboard (OB) spoilers. The diagram was

automatically generated from the PREMISE model and its detailed version helped the engineers to trace if every required function was satisfied and if every system component was actually needed. The upper part of the figure illustrates the control chain from one of the pilots' side sticks via control computers and the left-hand electro-hydraulic servo actuators (EHS) to the aileron. The solution part of the EFCS model was complemented by state machines to represent the behavior of the actuators. Besides expected nominal states, failure states were also considered, which enabled the engineers to perform a safety assessment based on a generated Fault Tree diagram[17].

All diagrams were automatically generated from the PREMISE model by DLR's "Parametric Architecture Designer" tool. This automatism improves the design efficiency as the engineers do not have to draw diagrams manually. Besides generating diagrams, the tool is capable of exporting different file formats for engineering tools such as SimMechanics (Mathworks) and FAULTTREE+ (Isograph), as already presented[15,16]. In this way, tool interoperability is made possible which contributes to a seamless transition to next design stages. By now, the tool supports the PARADIGMshift method completely. The model-based character and the way forward to a seamless transition to next development stages were shown as required in section 2.5. The example presented the application of the function-based approach by defining the flight control functions prior to the system elements. This methodical top-down proceeding enhances the chances for completeness of the functions tree and increased the designers' confidence related to the design quality. Additionally, it showed that the method considers graphical representations for different diagrams as required.

5. Conclusions

The conception stage is a trend-setting and therefore important stage for the development of systems. However, the effort to learn new tools is avoided by using commonly known but unspecialized tools such as Excel, which constrain the use of specialized design methods for feasibility studies. To overcome this issue, a suitable design method has to be defined first.

General cross-cutting methods for system development were discussed regarding their applicability for feasibility studies. The considered methods include simulation-, model-, and function-based approaches. A design method called PARADIGMshift was introduced integrating such beneficial concepts in a pragmatic and time-saving way. The method is divided into the parts problem domain, solutions domain, and trade-off analysis. Thinking of what the system will do is separated from how it will do it and from which solution may be the best. For every part, a pragmatic way of use is proposed, e.g. describing the system's behavior by state machines discretely instead of by extensive simulations dynamically.

Application examples from space and aerospace domains show the successful implementation of the method. The method is dependent on an Integrated Data Model like DLR's PREMISE model and should be supported by a design tool like DLR's "Parametric Architecture Designer".

References

1. ISO/IEC 15288. Systems and software engineering - System life cycle processes. 2008.
2. Haskins C (ed.). Systems Engineering Handbook v3.2.2. International Council on Systems Engineering (INCOSE); 2011.
3. Jacobsen I, Booch G, Rumbaugh J. UML - The Unified Software Development Process. Addison-Wesley; 1999.
4. Akao Y. QFD: Quality Function Deployment - Integrating Customer Requirements into Product Design. Taylor & Francis; 2004.
5. Bogus SM, Molenaar KR, Diekmann JE. Concurrent Engineering Approach to Reducing Design Delivery Time. J of Construction Engineering and Management; **131**:11. ASCE; 2005. p. 1179-1185.
6. Romberg O et al. Status of the Concurrent Engineering Facility at DLR Bremen. Deutscher Luft- und Raumfahrtkongress. Darmstadt: DGLR; 2008.
7. Systems Engineering Fundamentals. Defense Acquisition University Press; 2001.
8. Friedenthal S, Moore A, Steiner R. A Practical Guide to SysML. Morgan Kaufmann; 2011.
9. Karsai G et al. Design patterns for open tool integration. Software and Systems Modeling; **4**:2. Springer; 2005. p. 157-170.
10. Systems Engineering Vision 2020. International Council on Systems Engineering (INCOSE); 2007.
11. Yannakakis M. Hierarchical State Machines. Lecture Notes in Computer Science; **1872**. Springer; 2000. p. 315-330.
12. Witte L et al. The Mobile Asteroid Surface Scout (MASCOT) - System & Mission Engineering and Surface Operations Concept. Global Space Exploration Conference (GLEX). Washington D.C.: 2012.

13. Schumann H et al. Concurrent Systems Engineering in Aerospace: From Excel-based to Model Driven Design. In: Proceedings of 8th Conference on Systems Engineering Research (CSER). Hoboken: 2010. p. 685-694.
14. Schumann H, Berres A. Modellbasierter Systementwurf mit dem PrEMISE-Modell. In: M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2011. München: Carl Hanser; 2011. p. 35-44. Available in German.
15. Berres A, Schumann H. Closing the safety process gap: Early integration of safety assessment methods into systems engineering. In: M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2014. München: Carl Hanser; 2014. p. 143-152.
16. Schumann H, Zamov P, Escher S. Model-Based Design and Tool Data Exchange in Aerospace: A Case Study. CEAS Aeronautical Journal; **2**:1-4. Springer; 2011. p. 295-303.
17. Schumann H et al. Modell-basierter Systementwurf: Generierung von Fault-Trees. Deutscher Luft- und Raumfahrtkongress. Berlin: DGLR; 2012. English translation by H. Schumann.
18. Lamm JG, Weilkiens T. Funktionale Architekturen in SysML. In: M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010. München: Carl Hanser; 2010. p. 109–118. English translation by J. Lamm.